IdPUnsolicitedSSO

Usually in Shibboleth, the flow is assumed to be an SP requesting authentication by redirecting the client to the IdP, and then getting back a response. In the original SAML 1.0 and SAML 1.1 standards, though, SSO was described in only semi-interoperable terms as a response from the IdP to the SP, and the "request" portion was left out. This was carried over into SAML 2.0 as a mode called "IdP-initiated" or "unsolicited" SSO.

While this approach lacks interoperability, it has perceived benefits for some service providers; they get to do less work and push that work onto users and IdPs. So it isn't unusual to find SPs that refuse to support the standard fully and insist on this approach.

What is misunderstood about this feature is that it is **not** interoperable, despite being part of the standard. Interoperability requires a well-defined message, and the basic idea behind IdP-initiated SSO is that the message is up to the IdP. **Something** has to initiate the process, it can't magically start for no reason. So there is a request to the IdP, but it isn't a SAML message and no two IdPs are likely to work the same way. It's a proprietary mechanism.

SAML 1.x

In the original Shibboleth versions, the lack of a request message in SAML 1.x was supplemented with a simple request format defined in the Shibboleth Protocol Specification. Simply defined, it's a redirect with a small set of parameters:

- `providerId`
  - the name (i.e., the entity ID) of the service provider.
- `shire`
  - the URL of the response location at the SP (called the "Assertion Consumer Service")
- `target`
  - the target resource at the SP, or a state token generated by an SP to represent the resource.
- `time` (optional)
  - A timestamp to help with stale request detection.

Formally speaking, this is **not** IdP-initiated SSO; it's a proprietary request to the IdP that results in a response to the SP. If you refer back to the initial discussion above, you can see that that's actually the definition of IdP-initiated SSO.

SAML 2.0

Despite the fact that the SAML 2.0 standard requires this feature, the IdP prior to version 2.3.0 does not support an equivalent mechanism for SAML 2.0 SSO responses. The need to define a proprietary extension, plus a variety of other flaws in this whole concept, made us reluctant to support the feature. This caused a lot of problems, and resulted in a number of extensions and modifications to get the feature to work.

We decided to adapt the original protocol extension so that it can be used to trigger SAML 2.0 SSO in addition to legacy SAML 1.x responses. The two protocols are **not** supported at the same endpoint; that is, you can't send the request to one endpoint and expect the IdP to figure out which protocol to use. It's simply an alternative request format that requires the identified SP to support SAML 2.0.

Out of the box, this endpoint can be found at **/idp/profile/SAML2/Unsolicited/SSO** and the following parameters can be used:

- `providerId`
  - Name of the service provider.
- `shire` (optional)
  - URL of the response location at the SP (the "Assertion Consumer Service"), but can be omitted in favor of the IdP picking the default endpoint location from metadata. (See the SAML 2.0 Metadata spec for a precise definition of *default endpoint*.)
- `target` (optional)
  - Corresponds to `RelayState` in the SAML 2.0 protocol, but can be omitted.
- `time` (optional)
  - A timestamp to help with stale request detection.

As you can see, this is the same protocol as in the old days, but with more optional parameters, reflecting how we would have designed the protocol if we were starting from scratch today. Protocol syntax is compatible with the original so that existing links can be easily adapted and used, despite the fact that the terminology is outdated.

Configuration Issues

The default configuration files for Shibboleth IdP 2.3.0 and later need no further changes to use IdP-initiated SSO. To modify older configuration files to add support for IdP-initiated SSO after upgrading the IdP to IdP 2.3.0 or later, add the following profile handler in handler.xml:

```
<ProfileHandler xsi:type="SAML2SSO"
    inboundBinding="urn:mace:shibboleth:2.0:profiles:AuthnRequest"
    outboundBindingEnumeration="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign
                                urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
                                urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact">
  <RequestPath>/SAML2/Unsolicited/SSO</RequestPath>
</ProfileHandler>
```

Also add the following message decoder in internal.xml:

```
<entry>
  <key>
    <value>urn:mace:shibboleth:2.0:profiles:AuthnRequest</value>
  </key>
  <bean id="shibboleth.UnsolicitedSSODecoder"
        class="edu.internet2.middleware.shibboleth.idp.profile.saml2.UnsolicitedSSODecoder">
    <constructor-arg ref="shibboleth.IdGenerator"/>
  </bean>
</entry>
```

Note that the implementation of this feature is based on the addition of a custom MessageDecoder class (`UnsolicitedSSODecoder`) that accepts a request at the unsolicited endpoint location and passes control to the original SAML 2.0 SSO profile support. If a site wishes to support a customized protocol to add additional features or use different parameters, it's a simple matter to implement an alternative MessageDecoder and plug that in.

One caveat: a feature that few deployers would be relying on today is requiring that requests be signed, but that comes up in some higher security environments. In such a case, this feature will completely circumvent such a policy since it is essentially just an unsigned request. Such deployments would need to disable the unsolicited SSO endpoint in the IdP by commenting out (or removing) the above elements in handler.xml and internal.xml.